

# Hybrid Mobile Applications - ICD0018

TalTech IT College, Andres Käver, 2021-2022, Fall semester

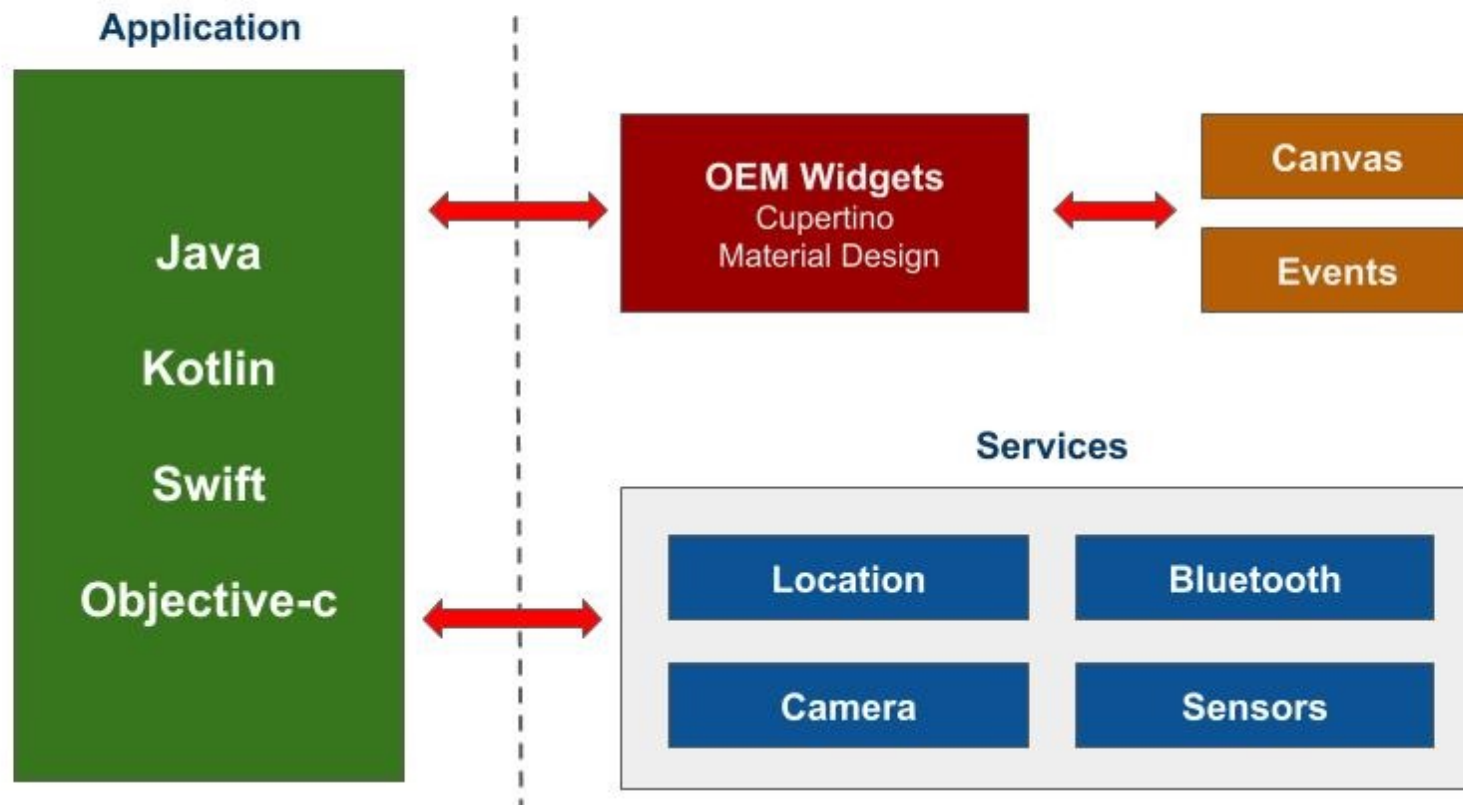
Email: [andres.kaver@taltech.ee](mailto:andres.kaver@taltech.ee)



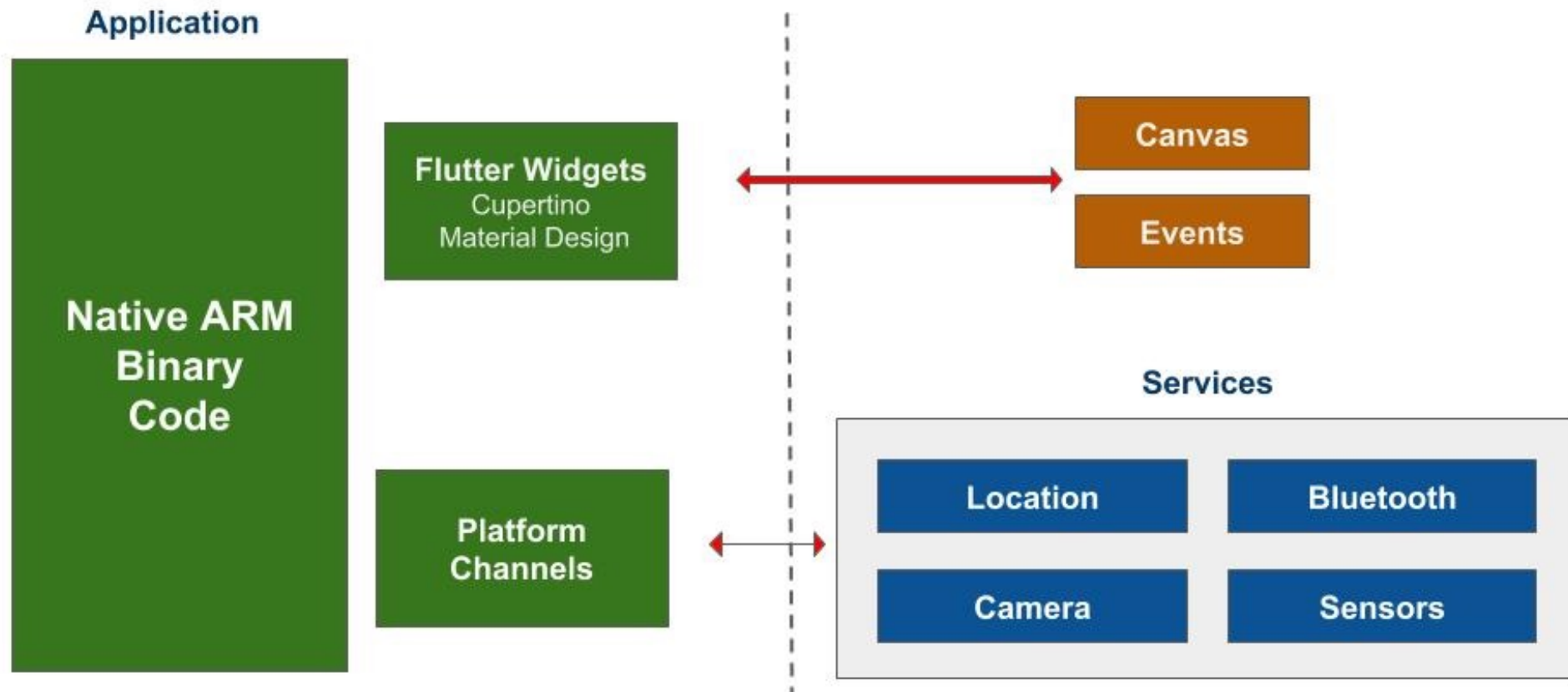
# Native SDKs

2

## OEM SDKs



## Flutter Approach



- ▶ Install (Xcode and/or Android studio required)
- ▶ Win
  - ▶ [https://storage.googleapis.com/flutter\\_infra\\_release/releases/stable/windows/flutter\\_windows\\_2.5.2-stable.zip](https://storage.googleapis.com/flutter_infra_release/releases/stable/windows/flutter_windows_2.5.2-stable.zip)
- ▶ macOS
  - ▶ [https://storage.googleapis.com/flutter\\_infra\\_release/releases/stable/macos/flutter\\_macos\\_2.5.2-stable.zip](https://storage.googleapis.com/flutter_infra_release/releases/stable/macos/flutter_macos_2.5.2-stable.zip)
- ▶ Linux
  - ▶ [https://storage.googleapis.com/flutter\\_infra\\_release/releases/stable/linux/flutter\\_linux\\_2.5.2-stable.tar.xz](https://storage.googleapis.com/flutter_infra_release/releases/stable/linux/flutter_linux_2.5.2-stable.tar.xz)

- ▶ Set up path variable  
export PATH="\$PATH:[PATH\_TO\_FLUTTER\_GIT\_DIRECTORY]/flutter/bin"
- ▶ Run
  - > flutter precache
  - > flutter doctor
- ▶ Fix all problems!

# Flutter - IDE

6

- ▶ Either VS Code or Android Studio
  - ▶ VS Code
    - ▶ Install Flutter extension
  - ▶ Android Studio
    - ▶ Install Flutter and Dart plugin
- ▶ Run again  
> flutter doctor

# Flutter – First app – VS Code

7

- ▶ **View > Command Palette.**
- ▶ Flutter: New Project
- ▶ Give name and location
- ▶ Let everything cool down, look for emulator connection on lower-right corner

Ln 1, Col 1   Spaces: 2   UTF-8   LF   Dart   Nexus 5X API 28 (android-x86 emulator)   😊   🔔 1

- ▶ Select Debug palette, click on Cog wheel
  - ▶ Create flutter debug configuration

DEBUG



No Configurations



# Flutter – VS Code

8

- ▶ Run your app
  - ▶ Debug > Start Debugging
  - ▶ Or press F5
- ▶ Wait for it!
- ▶ Try Hot Reload
  - ▶ Change some strings in main.dart
  - ▶ Save file
- ▶ Activate Dart Dev Tools when asked



# Flutter - Dart

9

- ▶ Flutter apps are written using the Dart programming language, also originally from Google and now an ECMA standard.
- ▶ Dart shares many of the same features as other modern languages such as Kotlin and Swift and can be trans-compiled into JavaScript code.
- ▶ As a cross-platform framework, Flutter most closely resembles React Native, as Flutter allows for a reactive and declarative style of programming.

# Flutter - Dart

10

- ▶ <https://dart.dev/>
- ▶ <https://dart.dev/guides/language/language-tour>
- ▶ Strongly typed, Single inheritance
- ▶ Mixins
- ▶ Implicit interfaces (every class is also an Interface)
- ▶ Implements for interfaces, extends for inheritance, with for mixin

# Flutter - Widget

11

- ▶ Everything is widget!
- ▶ Images, icons, and text in a Flutter app are all widgets. Even layout elements such as the rows, columns, and grids that arrange, constrain, and align other widgets, are widgets themselves.

```
import 'package:flutter/material.dart';

void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter',
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Flutter'),
        ),
        body: const Center(
          child: Text('Hello, World!'),
        ),
      ),
    );
  }
}
```

# Flutter - Widgets

12

- ▶ Widgets describe what their view should look like given their current configuration and state.
- ▶ When a widget's state changes, the widget rebuilds its description, which the framework diffs against the previous description in order to determine the minimal changes needed in the underlying render tree to transition from one state to the next.
- ▶ Widgets are subclasses mostly of either `StatelessWidget` or `StatefulWidget`, depending on whether your widget manages any state.
- ▶ Main task is to implement a `build()` function, which describes the widget in terms of other, lower-level widgets.
- ▶ Lowest widget is `RenderObjectWidget`

# Flutter - StatelessWidget

13

- ▶ User interface does not depend on anything other than the configuration information in the object/widget itself and the BuildContext in which the widget is inflated
- ▶ Stateless widgets receive arguments from their parent widget, which they store in final member variables. When a widget is asked to build(), it uses these stored values to derive new arguments for the widgets it creates.

```
class Frog extends StatelessWidget {  
  const Frog({  
    Key? key,  
    this.color = const Color(0xFF2DBD3A),  
    this.child,  
  }) : super(key: key);  
  
  final Color color;  
  final Widget child;  
  
  @override  
  Widget build(BuildContext context) {  
    return Container(color: color, child: child);  
  }  
}
```

# Flutter - StatefulWidget

14

- ▶ A widget that has mutable state.
- ▶ When the part of the user interface can change dynamically, e.g., due to having an internal clock-driven state, or depending on some system state.

# Flutter - StatefulWidget

15

```
class Bird extends StatefulWidget {  
  const Bird({  
    Key? key,  
    this.color =  
      const Color(0xFFFFE306),  
    required this.child,  
  }) : super(key: key);  
  
  final Color color;  
  final Widget child;  
  
  @override  
  _BirdState createState() =>  
    _BirdState();  
}
```

```
class _BirdState extends State<Bird> {  
  double _size = 1.0;  
  
  void grow() {  
    setState(() {  
      _size += 0.1;  
    });  
  }  
  
  @override  
  Widget build(BuildContext context) {  
    return Container(  
      color: widget.color,  
      transform: Matrix4  
        .diagonal3Values(_size, _size, 1.0),  
      child: widget.child,  
    );  
  }  
}
```

# Flutter - StatefulWidget

16

- ▶ Widget constructors only use named arguments.
- ▶ Named arguments can be marked as required using `@required`.
- ▶ The first argument is `key`, and the last argument is `child`, `children`, or the equivalent.



# Flutter - Basic widgets

17

- ▶ Text
  - ▶ The Text widget lets you create a run of styled text within your application.
- ▶ Row, Column
  - ▶ These flex widgets let you create flexible layouts in both the horizontal (Row) and vertical (Column) directions. The design of these objects is based on the web's flexbox layout model.

# Flutter - Basic widgets

18

## ► Stack

- Instead of being linearly oriented (either horizontally or vertically), a Stack widget lets you place widgets on top of each other in paint order. You can then use the Positioned widget on children of a Stack to position them relative to the top, right, bottom, or left edge of the stack. Stacks are based on the web's absolute positioning layout model.

## ► Container

- The Container widget lets you create a rectangular visual element. A container can be decorated with a BoxDecoration, such as a background, a border, or a shadow. A Container can also have margins, padding, and constraints applied to its size. In addition, a Container can be transformed in three-dimensional space using a matrix.

▶ THE END

# Flutter

20

# Flutter

21

# Flutter

22

# Flutter

23

# Flutter

24

```
import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Flutter'),
        ),
        body: Center(
          child: Text('Hello, World!'),
        ),
      ),
    );
  }
}
```

► Lets add a Row

```
body: Center(
  child: Row(
    mainAxisAlignment:
      MainAxisAlignment.center,
    children: <Widget>[
      Text('0'),
      Text('0'),
      Text('0'),
    ],
  ),
),
```



- Move numbers to separate Widgets

```
class CustomTextContainer extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return Text('00');  
  }  
}
```

```
children: <Widget>[  
  CustomTextContainer(),  
  CustomTextContainer(),  
  CustomTextContainer(),  
],
```

# Flutter

26

# Flutter

27

# Flutter

28

# Flutter

29

# Flutter

30